

Реактивное программирование

управляем потоками данных
...и пусть события сами идут

Витя Русакович,
*компания “GP software.travel”,
Минск, Беларусь*

Ближайшие 40 минут:

2

- ~~Хайп~~
- ~~Angular~~
- ~~ReactJS~~
- ~~Redux~~

**Из чего состоит Front-
End?**

СОБЫТИЯ

**Где мы используем
события?
ВЕЗДЕ**

ПРЕДВАРИТЕЛЬНЫЙ СТАРТ

СОБЫТИЕ



АКТИВНОСТЬ
ФАЗА

СОБЫТИЕ



Что не так с событиями?

- как их объединять?
- как их фильтровать?
- как собирать данные от разных событий?
- и не сойти с ума от асинхронности
- не строить Пирамиды Обратных Вызовов?
- ... (еще примерно 42 пункта)

O, ужасный callback!

```
$('#some-element').animate({ width: 100, height: 50 }, {
  duration: 300,
  complete: function () {
    // Step 2:
    $('#another-element').fadeOut(300, function () {
      // Step 3:
      setTimeout(function () {
        // Step 4:
        $.ajax({ url: 'http://google.com' })
          .done(function () {
            // Step 5:
            setTimeout(function () {
              // Step 6:
              $('#third-element').remove();
            }, 250);
            });
          }, 300);
    });
  }
});
```

O, ужасный callback!

```
step1(function (value1) {  
    step2(value1, function(value2) {  
        step3(value2, function(value3) {  
            step4(value3, function(value4) {  
                // Do something with value4  
            });  
        });  
    });  
});  
});
```


O, ужасный callback! (и доступ к данным)

9

```
step1(function (value1) {  
    step2(value1, function(value2) {  
        step3(value2, function(value3) {  
            if (value2 === 4) return null  
            step4(value3, function(value4) {  
                if (value2 > value3) work()  
            });  
        });  
    });  
});
```

Варианты решения: Promise

10

```
new Promise((resolve, reject) => {})  
  .then(step2)  
  .then(step3)  
  .then(step4)  
  .then(value4 => alert(value4))  
  .catch(error => console.warn(error));
```

Возможности Promise

`Promise.all()`

`Promise.prototype.catch()`

`Promise.prototype.then()`

`Promise.race()`

`Promise.reject()`

`Promise.resolve()`

Возможности Promise

12

Promise.all()

Promise.prototype.catch()

Promise.prototype.then()

Promise.race()

Promise.reject()

Promise.resolve()

2012 год



Windows® 8.1

Reactive Extensions for JS

...is a library to compose asynchronous and event-based programs using observable collections and LINQ-style query operators.

LINQ - Language-INtegrated Query

15

```
string searchTerm = "data";  
string[] source = text.Split(...);  
// Create the query. .ToLowerInvariant == .toLowerCase()  
var matchQuery = from word in source  
    where word.ToLowerInvariant() == searchTerm.ToLowerInvariant()  
    select word;  
// Count the matches, which executes the query.  
int wordCount = matchQuery.Count();
```

Реактивные расширения

...библиотека для создания асинхронных и основанных на событиях программ используя наблюдаемые коллекции и LINQ операторы.

Реактивное программирование

17

- парадигма, ориентированная на потоки данных и распространение изменений.

Это означает, что должна существовать возможность легко выражать статические и динамические потоки данных, а также то, что выполняемая модель должна автоматически распространять изменения сквозь поток данных.



ВИКИПЕДИЯ
Свободная энциклопедия

Реактивное программирование

18

- парадигма, ориентированная на **потоки данных** и распространение изменений.

Это означает, что должна существовать возможность легко выражать статические и динамические **потоки данных**, а также то, что выполняемая модель должна **автоматически** распространять изменения сквозь поток данных.





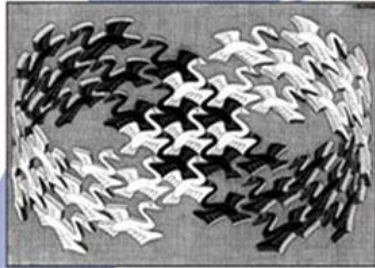
Reactive Extensions

an API for asynchronous programming
with **observable** streams

Design Patterns

Elements of Reusable Object-Oriented Software

Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides



Cover art © 1994 M.C. Escher / Cordon Art - Baarn - Holland. All rights reserved.

Foreword by Grady Booch



ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES

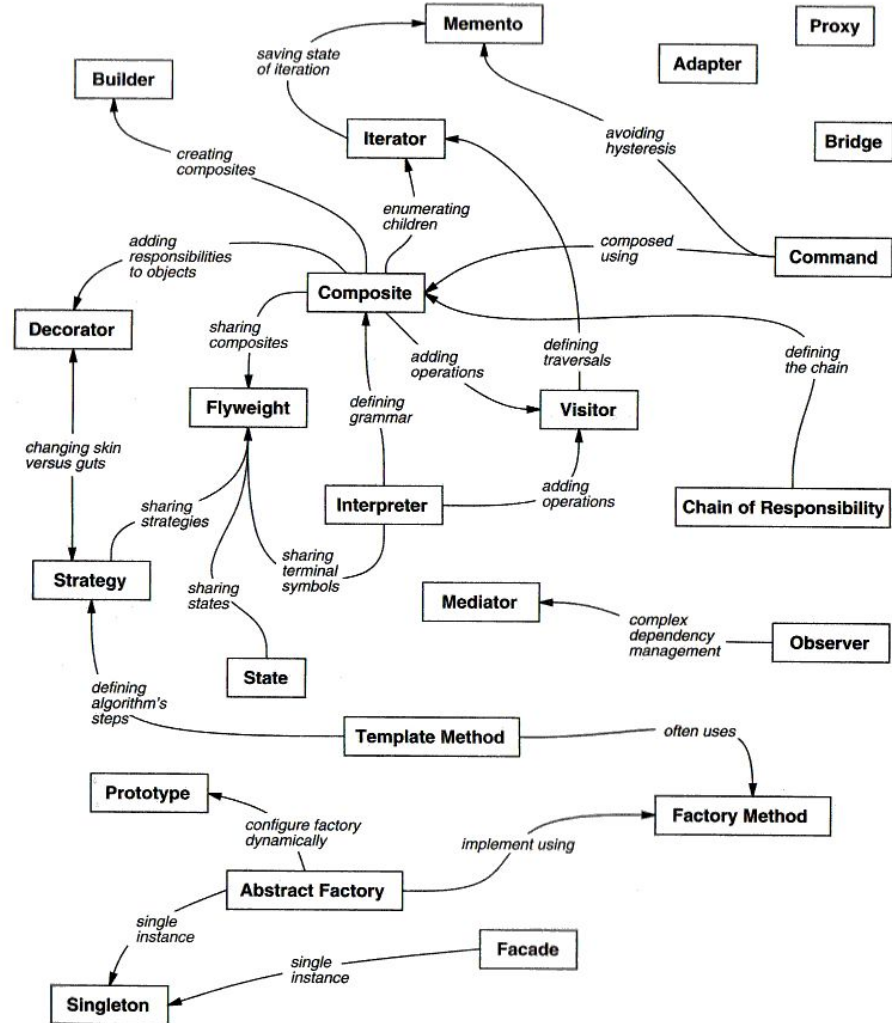
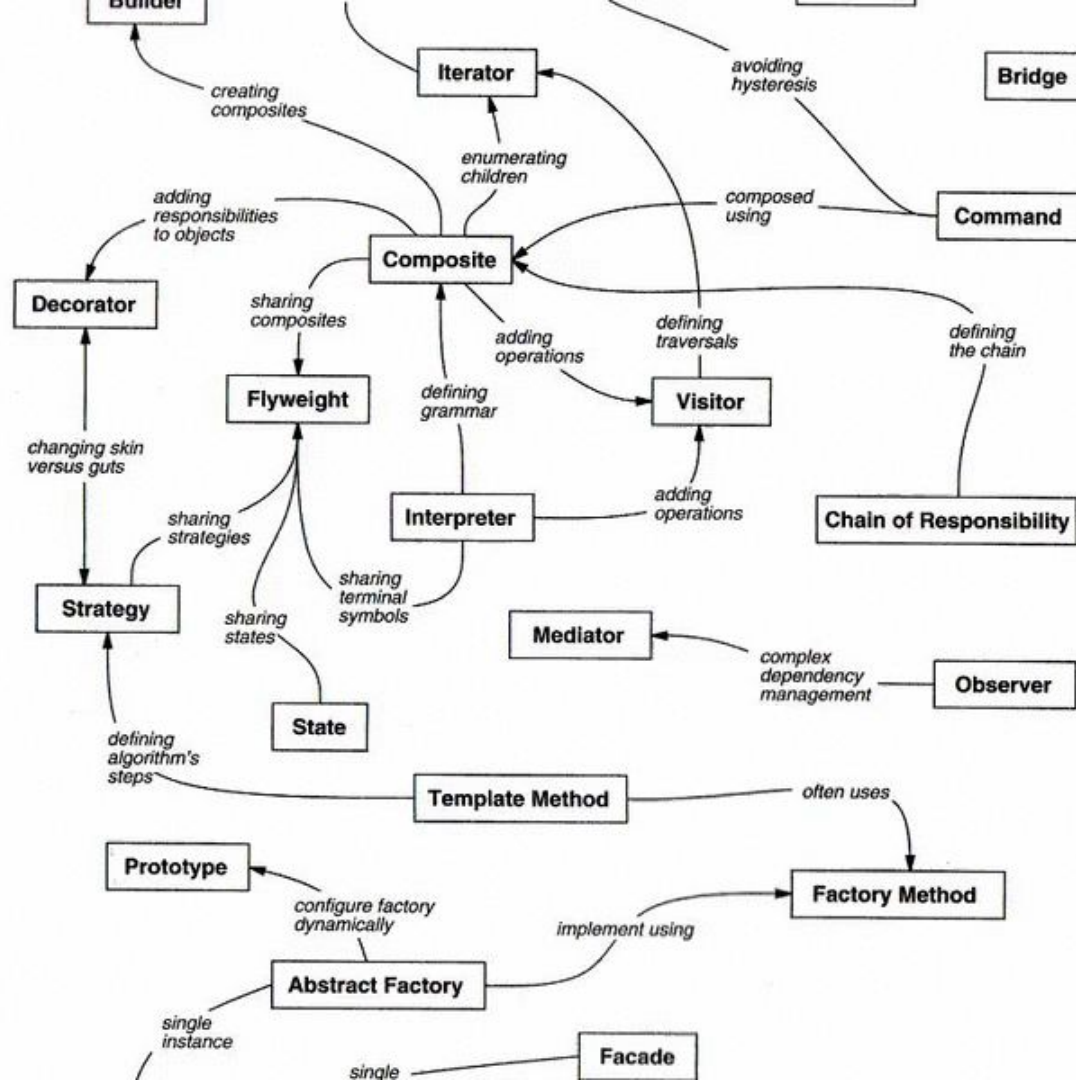


Figure 1.1: Design pattern relationships



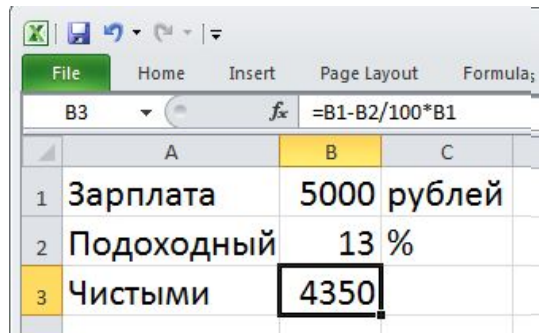
The Reactive Manifesto

<http://www.reactivemanifesto.org/>

Реактивность каждый день

23

Excel



The screenshot shows an Excel spreadsheet with the following data:

	A	B	C
1	Зарплата	5000 рублей	
2	Подходный	13 %	
3	Чистыми	4350	

The formula bar at the top shows the formula $=B1-B2/100*B1$ for cell B3.

$$=B1-B2/100*B1$$

Второй пример жизненной *реактивности*: 24

Ищем работу:

● Ищу надомную работу –
не грибы. 8 91607044

обзванивать вакансии

звонить

Не реактивно



разместить резюме

ждать звонков

Реактивно



Состав RxJS:

25

- объекты для представления асинхронных потоков данных
- операторы для создания, фильтрации и управления подобными объектами и данными внутри них
- немного магии

Основа идеологии

Наблюдатель

Observer

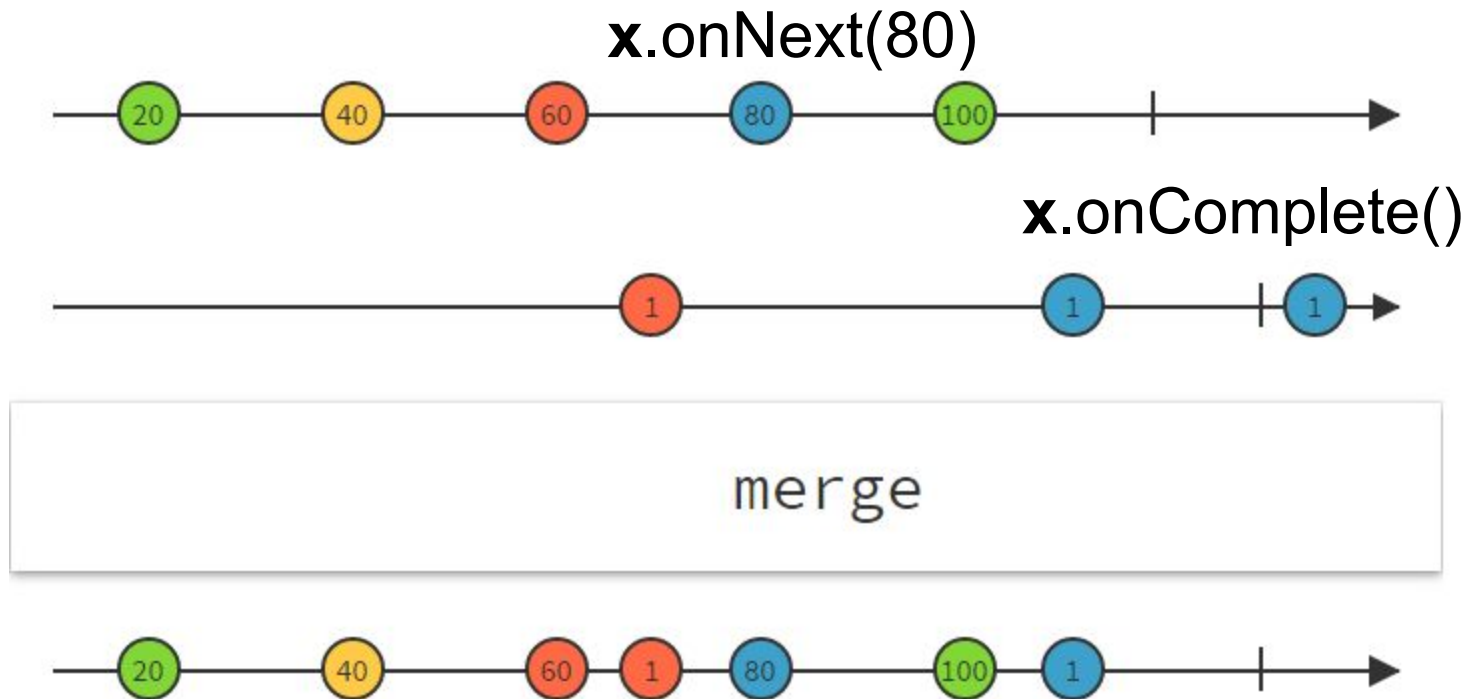
Наблюдаемое (поток)

Observable

подписывается (*subscribe*) \Rightarrow
 \Leftarrow присылает сообщения (*onNext*)

Observable sequence

27



`z = x.merge(y)` или `z = Rx.Observable.merge(x, y)`

Используем привычные функции

Array

- concat
- every
- filter
- map
- reduce
- some
- **forEach**

query

Observable

- concat
- all
- where
- select
- aggregate
- any
- **subscribe**

Новый источник событий - click

29

```
const input = document.getElementById('input');  
const source = Rx.Observable.fromEvent(input, 'click');  
const subscription = source.subscribe(  
  x => console.log('Next: Clicked!'),      // .onNext()  
  err => console.log('Error: ', err)        // .onError()  
  () => console.log('Completed')           // .onComplete()  
);  
input.trigger('click');  
// => Next: Clicked!
```



Новый источник событий - Promise

30

```
const promise = Promise.reject(new Error('reason'));  
const source = Rx.Observable.fromPromise(promise);  
const subscription = source.subscribe(  
  x => console.log('Next msg')           // .onNext()  
  err => console.log('Error: %s', err)    // .onError()  
  () => console.log('Completed')          // .onComplete()  
);
```

// => Error: Error: reject



Новый источник событий - Promise

31

```
// Create a promise which resolves 42
var promise = Promise.resolve(42)
var source = Rx.Observable.fromPromise(promise);
const subscription = source.subscribe(
    x => console.log('Next msg')           // .onNext()
    err => console.log('Error: %s', err)    // .onError()
    () => console.log('Completed')         // .onComplete()
);
// => Next: 42
// => Completed
```



Поддержка родных событий jQuery 32

- `bind` - `bindAsObservable`
- `delegate` - `delegateAsObservable`
- `live` - `liveAsObservable`
- `on` - `onAsObservable`
- `$.Deferred.toObservable` / `Rx.Observable.toDeferred`

Мышь и клавиатура

- `change` - `changeAsObservable`
- **`click` - `clickAsObservable`**
- `dblclick` - `dblclickAsObservable`
- `focus` - `focusAsObservable`
- `focusin` - `focusinAsObservable`
- `focusout` - `focusoutAsObservable`
- `hover` - `hoverAsObservable`
- `keydown` - `keydownAsObservable`
- `keypress` - `keypressAsObservable`
- `keyup` - `keyupAsObservable`
- `load` - `loadAsObservable`
- `mousedown` - `mousedownAsObservable`
- `mouseenter` - `mouseenterAsObservable`
- `mouseleave` - `mouseleaveAsObservable`
- `mousemove` - `mousemoveAsObservable`
- `mouseout` - `mouseoutAsObservable`
- `mouseenter` - `mouseenterAsObservable`
- `mouseleave` - `mouseleaveAsObservable`
- `mousemove` - `mousemoveAsObservable`
- `mouseout` - `mouseoutAsObservable`

\$.ajax и анимация

- **ajax** - ajaxAsObservable

- **get** - getAsObservable
- **getJSON** - getJSONAsObservable
- **getScript** - getScriptAsObservable
- **post** - postAsObservable

- **animate** - animateAsObservable

- **fadeIn** - fadeInAsObservable
- **fadeOut** - fadeOutAsObservable
- **fadeTo** - fadeToAsObservable
- **fadeToggle** - fadeToggleAsObservable
- **hide** - hideAsObservable

- **show** - showAsObservable

- **slideDown** - slideDownAsObservable
- **slideToggle** - slideToggleAsObservable
- **slideUp** - slideUpAsObservable

Состав RxJS:

35

- объекты для представления асинхронных потоков данных
- **операторы** для создания, фильтрации и управления подобными объектами и данными внутри них
- немного магии

Операторы объединения

- Amb
- Merge
- Concat
- CombineLatest
- Zip
- Repeat

Оператор объединения - combineLatest 37

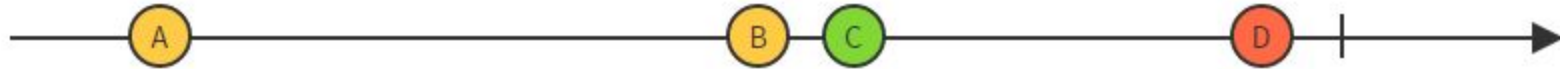
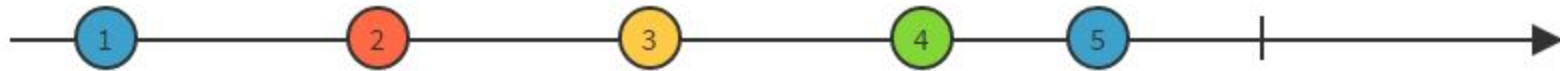


```
combineLatest((x, y) => "" + x + y)
```

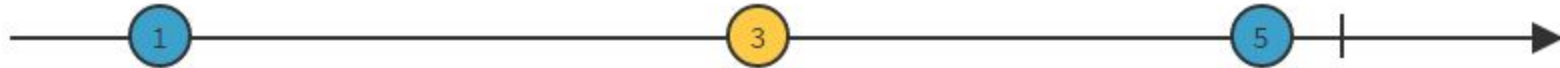


Оператор объединения - sample

38

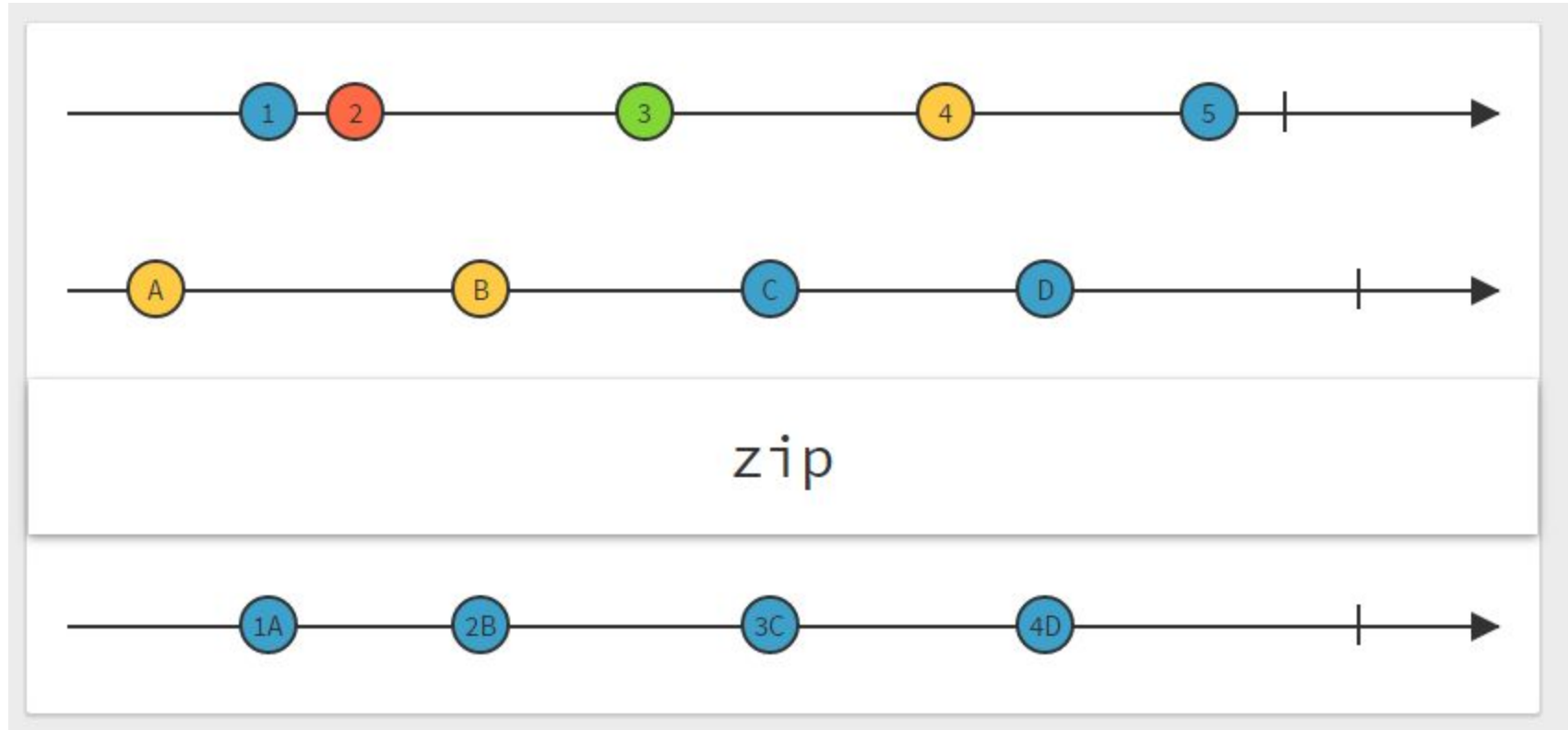


sample



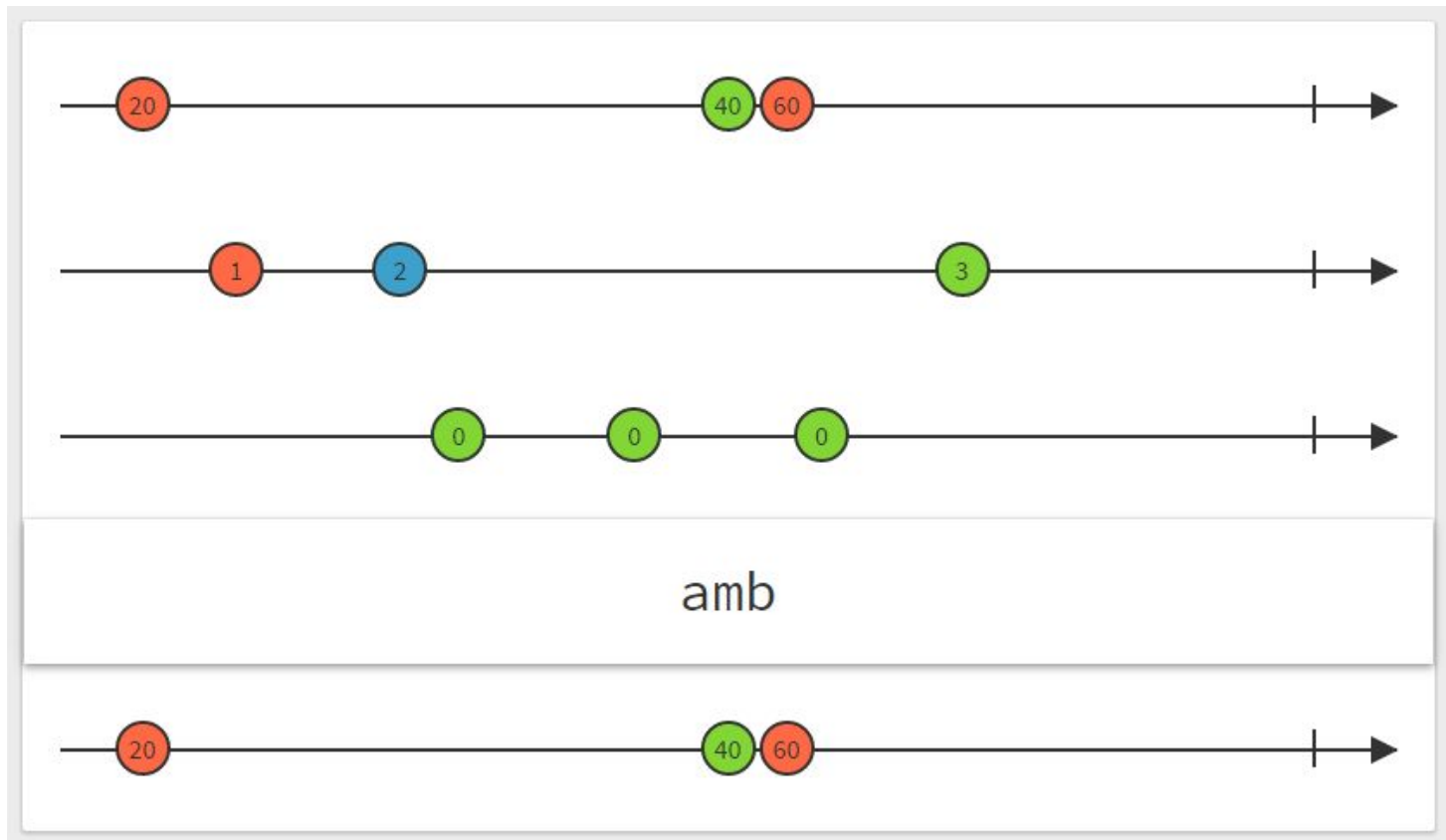
Операторы объединения - zip

39



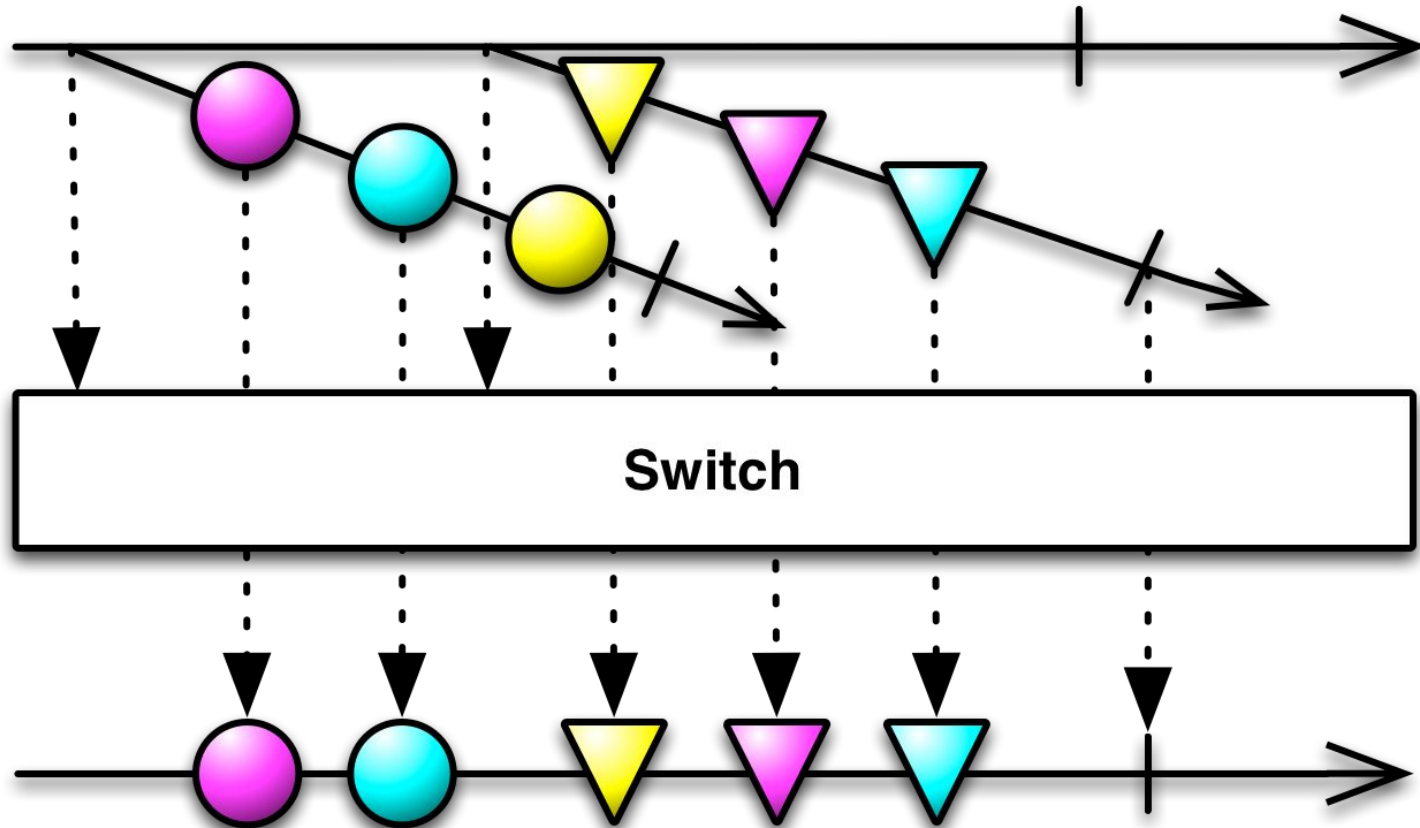
Условный оператор - amb

40



Оператор переключения - switch()

41



Операторы фильтрации и выбора

42

- `where (filter)` / `whereTrue` / `whereFalse`
- `take` / `takeUntil` / `takeWhile`
- `skip` / `skipUntill` / `skipWhile`
- `select (map)` / `selectMany` / `selectProperty`

Фильтрация и выбор - filter/where

43



```
filter(x => x > 10)
```



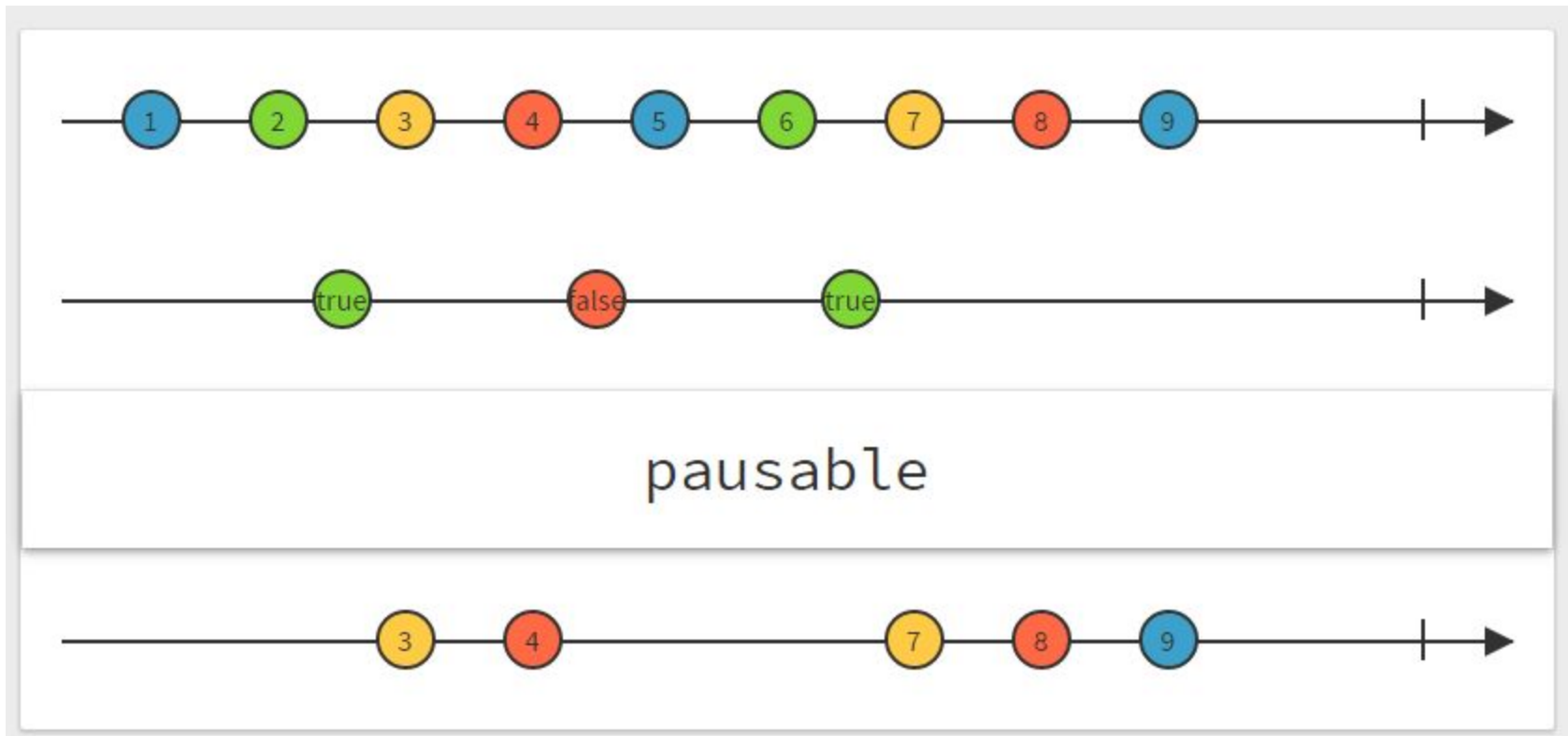
Операторы для агрегации данных

44

- scan
- count
- min
- max
- groupBy
- distinct / distinctUntilChanged
- throttle

Операторы для данных - pausable

45



pausableBuffered

46

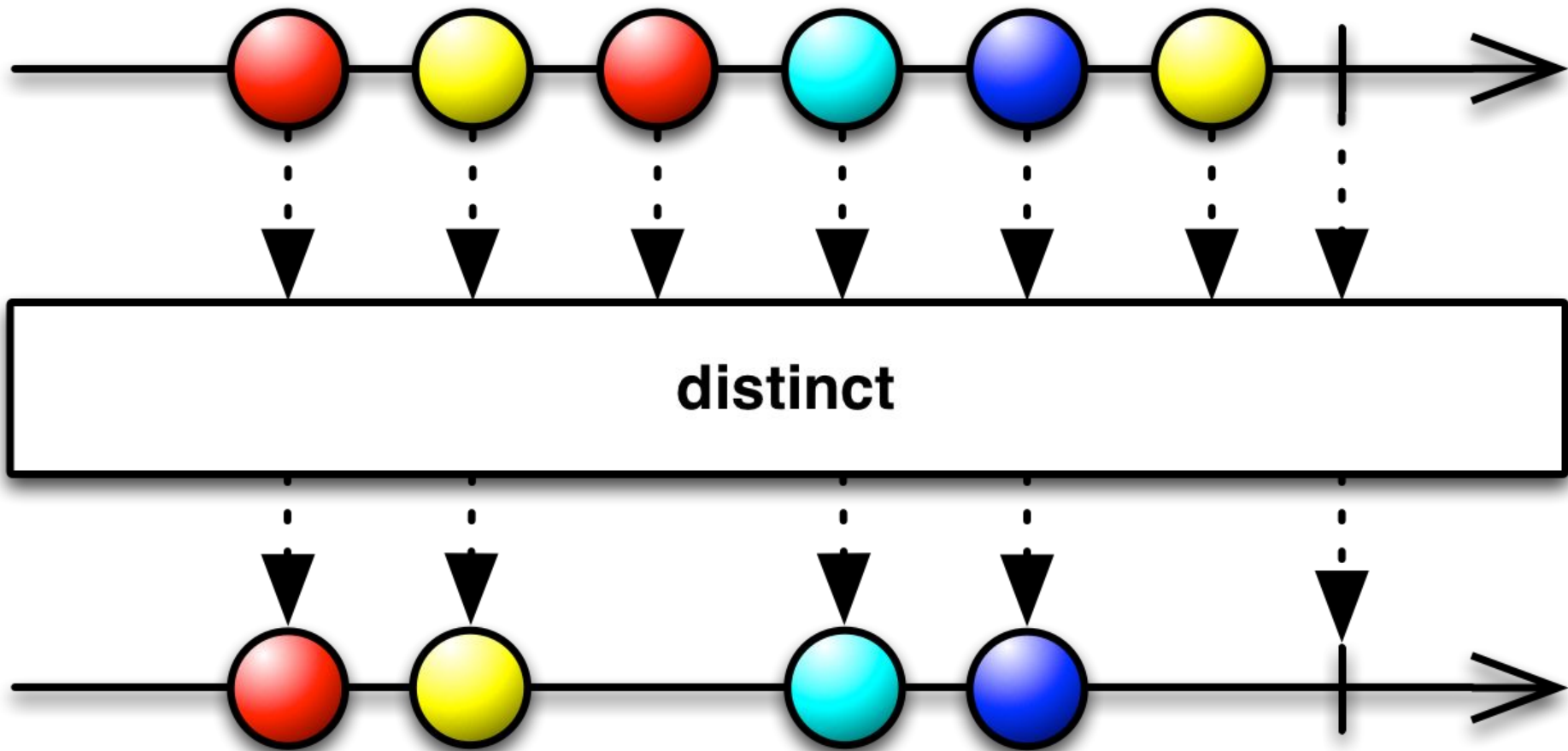


pausableBuffered



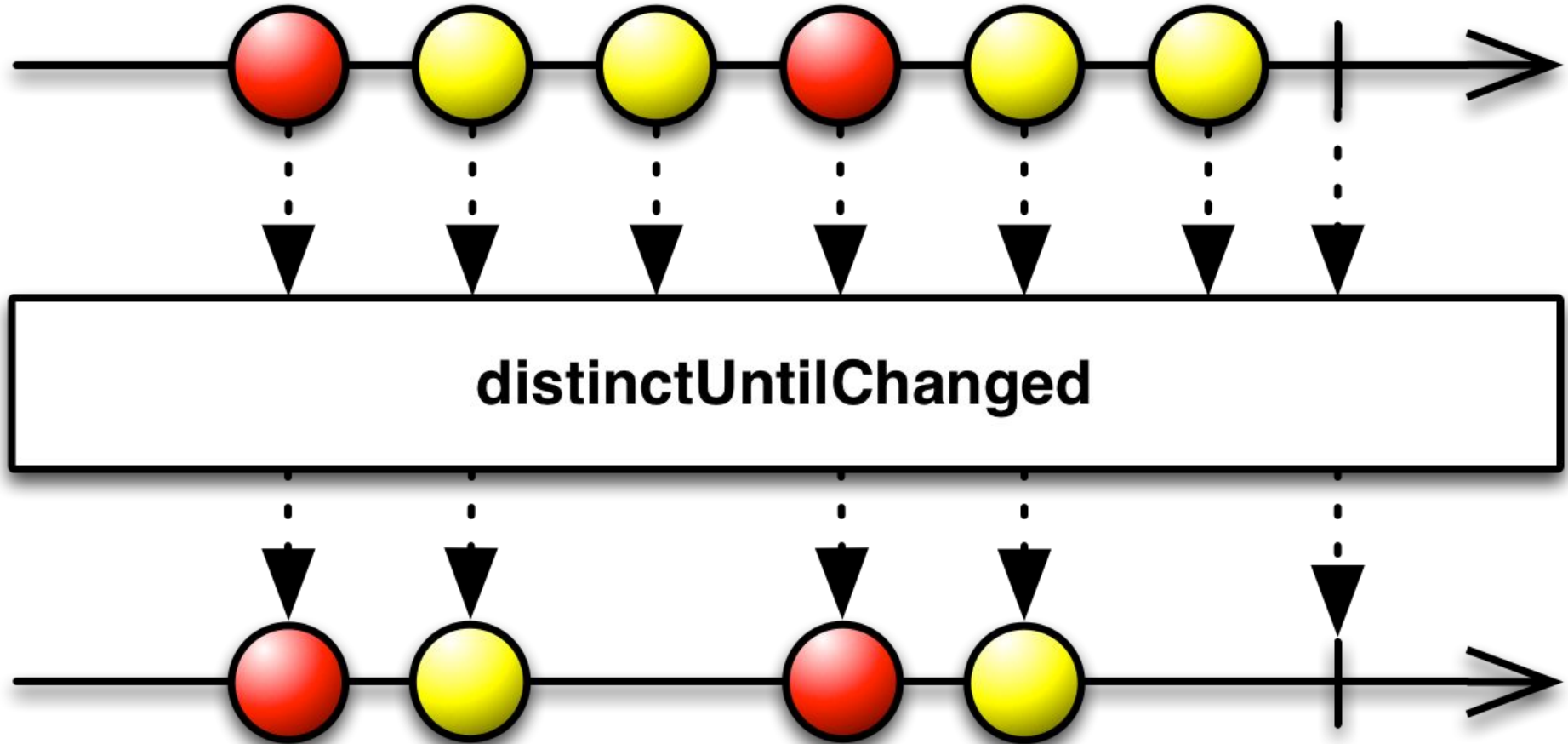
distinct

47



distinctUntilChanged

48



Операторы - over 9000

aggregate	every	minBy	subscribe
all	expand	multicast	subscribeOn
amb	filter	never	sum
and	finally finallyAction	observeOn	switch switchLatest
any	find	onErrorResumeNext	take
asObservable	findIndex	onErrorResumeNext	takeLast
average	first	pluck	takeLastBuffer
buffer	firstOrDefault	publish	takeLastBufferWithTime
bufferWithCount	flatMap	publishLast	takeLastWithTime
bufferWithTime	flatMapLatest	publishValue	takeUntil
bufferWithTimeOrCount	for forIn	range	takeWhile
case switchCase	forkJoin	reduce	throttle
catch catchException	fromArray	refCount	throttleWithSelector
catch catchException	generate	repeat	throw throwException
combineLatest	generateWithAbsoluteTime	repeat	timeInterval
concat	generateWithRelativeTime	replay	timeout
connect	groupBy	retry	timeoutWithSelector
contains	groupByUntil	return returnValue	timer
count	groupJoin	sample	timestamp
create	if ifThen	scan	toArray
createWithDisposable	ignoreElements	select	toAsync
defaultIfEmpty	interval	selectMany	using
defer	isEmpty	selectSwitch	when
delay	join	single	where
delayWithSelector	last	singleOrDefault	while whileDo
dematerialize	lastOrDefault	skip	window
distinct	manySelect	skipLast	windowWithCount
distinctUntilChanged	map	skipLastWithTime	windowWithTime
do doAction	max	skipUntil	windowWithTimeOrCount
doWhile	maxBy	skipWhile	zip
elementAt	merge	some	
elementAtOrDefault	mergeObservable	start	
empty	min	startWith	

Языки с библиотеками RX

50

1. node.js
2. Java
3. Ruby
4. Python
5. ObjectiveC
6. C++
7. RxKotlin
8. <https://github.com/Reactive-Extensions>

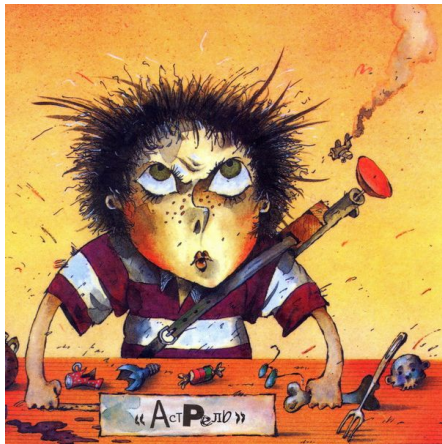
Демо

закрепим полученные знания

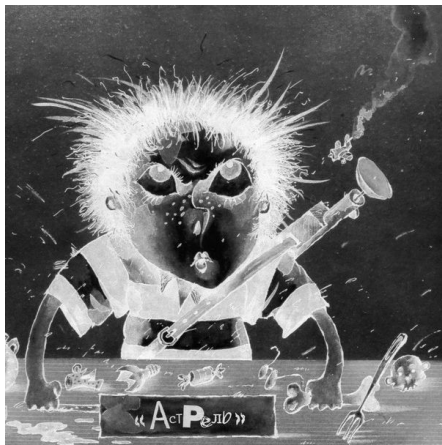
Вредные советы

52

- одноразовые потоки
- потоки-сироты (“*subscribe? He, не слышал*”)
- превращаем все события в потоки, все!
- общие названия потоков (*dataStream*)
- свои операторы не читая документацию



- простые данные в сообщениях
- говорящие названия для потоков
- используем статические методы
- промежуточные потоки
- функциональный подход



Полезные советы

54

простые данные в сообщениях

```
Rx.Observable.fromEvent(input, 'click')  
  .map(isClickInside) // Boolean vs MouseEvent  
  .where(flag => flag)  
  .subscribe(sideEffect)
```

```
function isClickInside(ev) { return ev.offsetX > 0 }
```

Полезные советы

используем статические методы:

zip, merge, combine

```
const threeStreams = stream1
```

```
  .merge(stream2)
```

```
  .merge(stream3)
```

```
const threeStreams = Rx.Observable.merge(  
  stream1, stream2, stream3
```

```
)
```

Полезные советы

промежуточные потоки

```
var streamPanelContentLoaded = clickButtonObservable
    .doAction(togglePanelLoading) // UI
    .select(loadPanelContent)     // ajax
    .switchLatest()               // flatten
    .doAction(togglePanelLoading) // UI

function togglePanelLoading(bookingModel) {
    $('#' + bookingModel.getId()).toggleClass('loading')
}
```


Полезные советы

промежуточные потоки

```
var streamPanelContentLoaded = ...
```

```
streamPanelContentLoaded
```

```
    .combineLatest(userBalanceUpdateStream, _.merge)  
    .subscribe(panelContentLoaded) //render schedule + balance
```

```
var streamCruiseSelection = streamPanelContentLoaded  
    .selectProperty('bookingModel')  
    .where(bookingModel => bookingModel.isCruise)
```

Полезные советы

58

свои операторы `.log()` и `selectAlways()`

```
Rx.Observable.prototype.log = function(message) {  
  this.subscribe(x => console.log(message, x))  
  return this  
}
```

```
Rx.Observable.prototype.selectAlways = function(value) {  
  return this.select(function() { return value })  
}
```

Полезные советы

свои операторы `.log()` и `.selectAlways()`

```
var piterIsSunToday = loadClick
    .select(gismeteoAjaxStream("Piter")).switch()
    .select(w => !w.clouds)
```

loadClick

```
.selectAlways(true) // .select(() => true)
.log("clicked")      // .tap(() => console.log("clicked"))
.subscribe(toggleSun)
```

Не понравился RxJS?

60



BaconJS



```
var up = $('#up')
  .asEventStream('click')
var down = $('#down')
  .asEventStream('click')
```

```
var counter = up.map(1)
  .merge(down.map(-1))
  .scan(0, (x,y) => x + y)
```

```
counter.assign($('#counter'), 'text')
```



0



KefirJS



```
var up = Kefir.fromEvents(
  $('#up'), 'click')
var down = Kefir.fromEvents(
  $('#down'), 'click')
```

```
var counter = up.map(() => 1)
  .merge(down.map(() => -1))
  .scan(0, (x,y) => x + y)
```

```
counter.onValue(
  value => $('#counter').val(value)
)
```

He top? Elm!

62

```
main = beginnerProgram { model = 0, view = view, update = update }
```

```
view model =
```

```
  div []
```

```
    [ button [ onClick Decrement ] [ text "-" ]
```

```
      , div [] [ text (toString model) ]
```

```
      , button [ onClick Increment ] [ text "+" ]
```

```
    ]
```

```
type Msg = Increment | Decrement
```

```
update msg model =
```

```
  case msg of
```

```
    Increment -> model + 1
```

```
    Decrement -> model - 1
```



1. <http://reactivex.io/documentation/observable.html>
2. <https://xgrommx.github.io/rx-book/>
3. egghead.io/series/introduction-to-reactive-programming
4. <https://habrahabr.ru/company/jugru/blog/302284/>
5. http://www.introtorx.com/Content/v1.0.10621.0/00_Foreword.html
6. <https://github.com/Reactive-Extensions>
7. <http://blogs.microsoft.co.il/blogs/bnaya/archive/2010/02/25/rx-for-beginners-toc.aspx>
8. <http://www.slideshare.net/mattpodwysocki/cascadiajs-dont-cross-the-streams>
9. <http://www.slideshare.net/panesofglass/rx-workshop>
10. <http://www.infoq.com/reactive-extensions/>
11. <http://mywebbasedcomputer.com/users/johngslater/tech/rx/bubbleDiagrams.html>
12. <https://github.com/Netflix/RxJava/wiki/Observable-Utility-Operators>
13. [Matthew Podwysocki](#)

Мой первый реактив

64

`Rx.Observable.from`

`Rx.Observable`

```
.from(document.querySelectorAll('a'))  
.where(el => el.innerText.indexOf('ok'))  
.subscribe(el => el.className = 'ok')
```



Спасибо! Вопросы?



Виктор Русакович из Минска, Беларусь
nemiga@gmail.com